# A

# Environment Description

The instructions bellow are for the development of the hardware implementation, the development of the software application and the programming of the bitstream and the application binary files into the fpga board. While the full development environment can be used in most linux distributions, this guide describes how to use a combination of Windows for the Vivado Design Suite, Windows Subsystem for Linux (WSL) for the application development, and Raspberry Pi OS to program the files into the board. The Raspberry Pi is only used to facilitate remote work, thus the board is connected to it.

## A.1 Installation of Software Tools

### A.1.1 Vivado Design Suite

1. Navigate to https://reference.digilentinc.com/vivado/installing-vivado/start

2. You will be guided to Xilinx's download page to download Vivado Design Suit

3. It is recommended to download the ”Self Extracting Web Installer”

4. The vivado installer will guide through the installation process

5. After Vivado and Vitis HLS are installed, it’s necessary to install the cable drivers (follow the digilent guide above)

### A.1.2 Zephyr Development Environment

1. Install VScode by following the link https://code.visualstudio.com/Download

2. Install pip: sudo apt install python3-pip

3. Install pyelftools: sudo apt-get install -y python3-pyelftools python-pyelftools

4. Install FuseSoC: sudo pip3 install –upgrade fusesoc

5. Install OpenOCD:
5.1 Install the required dependencies:

```
sudo apt-get install libusb-1.*
sudo apt-get install pkg-config
sudo apt-get install libtool
```

5.2 Clone the riscv-openocd github repository and install OpenOCD:

```
git clone https://github.com/riscv/riscv-openocd.git
cd riscv-openocd
./bootstrap
./configure --prefix=/opt/riscv --program-prefix=riscv- --enable-ftdi --enable-jtag_vpi
make
sudo make install
```

6 Install the required dependencies:

```
sudo apt install --no-install-recommends git cmake ninja-build gperf \
ccache dfu-util device-tree-compiler wget \
```

```
python3-dev python3-pip python3-setuptools python3-tk python3-wheel xz-utils file \
make gcc gcc-multilib g++-multilib libsdl2-dev libmagic1 \
```

**Note:** Make sure that the dependencies intalled are the correct versions according to this forum post.

7 Install cmake:

```
wget -O - https://apt.kitware.com/keys/kitware-archive-latest.a sc 2>/dev/null |
sudo apt-key add -
sudo apt-add-repository 'deb https://apt.kitware.com/ubuntu/ bionic main'
sudo apt update
sudo apt install cmake
```

8 Install west:

```
pip3 install --user -U west
echo 'export PATH=~/.local/bin:"$PATH"' >> ~/.bashrc
source ~/.bashrc
```

9 Install Zephyr SDK:

9.1 Download the 0.12.4 SDK installer

9.2 Run the installer:

```
chmod +x zephyr-sdk-0.12.4-x86_64-linux-setup.run
./zephyr-sdk-0.12.4-x86_64-linux-setup.run -- -d ~/zephyr-sdk-0.12.4
```

### A.1.3 FPGA Programming Environment

A Raspberry Pi board connected to the FPGA was used to facilitate remote work. This Raspberry Pi board was running the Raspberry Pi OS. On it, the RISC-V Toolchain and OpenOCD were installed so that the Raspberry Pi can program the SoC bitstream and the application's binary files. If the Raspberry Pi board is not bwing used some of the following steps may not be necessary.

1. Install the RISC-V Toolchain:

```
sudo apt-get install git autoconf automake autotools-dev curl \
libmpc-dev libmpfr-dev libgmp-dev gawk build-essential bison flex \
texinfo gperf libtool patchutils bc zlib1g-dev libexpat-dev
```

```
git clone --recursive https://github.com/riscv/riscv-gnu-toolchain
cd riscv-gnu-toolchain/
./configure --prefix=/opt/riscv --with-arch=rv32imc
sudo make
export PATH=$PATH:/opt/riscv/bin
```

2. Install OpenOCD:

```
sudo apt-get install libusb-1.*
sudo apt-get install pkg-config
git clone https://github.com/riscv/riscv-openocd.git
cd riscv-openocd
./bootstrap
./configure --prefix=/opt/riscv --program-prefix=riscv- --enable-ftdi --enable-jtag_vpi
make
sudo make install
```

## A.2 Developing the Bitstream and Binary Files

### A.2.1 Generation of the Bitstream

1. Open the rvfpga Vivado project in the Thesis github repository

2. Include two folder for the Pulp Platform. In the FLow Navigator pane click on *Settings*. In the opened window click on *General* and then next to the *Verilog Option* click on the three dots to select the directories. Navigate to the following directories and select the include foulder.

```
[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/AxiInterconnect/
    pulp-platform.org_axi_0.25.0/include

[RVfpgaPath]/RVfpga/src/OtherSources/pulp-platform.org_common_cells_1.20.0/include
```

3. Generate the bitstream by clicking on *Generate Bitstream* under the *Flow* dropdown menu. The bitstream will be created in the following path:

```
[RVfpgaPath]/RVfpga/rvfpga.runs/impl_1
```

### A.2.2 Generation of the Zephyr Application Binary

1. Create a new directory that will be the workspace to develop the Zephyr application. In this guide the workspace folder is called SweRVolf.

2. Navigate to the workspace directory and create an environment variable with the path to the workspace:

```
export WORKSPACE=\$(pwd)
```

3. Add the FuseSoC base library to the workspace:

```
fusesoc library add fusesoc-cores https://github.com/fusesoc/fusesoc-cores
```

4. Add the swrvolf library:

```
fusesoc library add swervolf https://github.com/chipsalliance/Cores-SweRVolf
```

**Note:** If fusesoc can't find the swervolf library swap the URL to https://github.com/chipsalliance/VeeRwolf.

5. Set the swervolf directory as a new environment variable:

```
export SWERVOLF_ROOT=$WORKSPACE/fusesoc_libraries/swervolf
```

6. Create a west workspace in the same directory as the fusesoc workspace:

```
west init
west config manifest.path fusesoc_libraries/swervolf
west update
```

7. Compile the Zephyr application: 7.1 Navigate to the the application samples directory:

```
$WORKSPACE/zephyr/samples
```

7.2 Copy the zephyr application folder from the Thesys repository to the samples folder.

7.3 Select what application to compile by uncommenting the target sources in the file *CMakeLists.txt*.

7.4 Compile the Zephyr application using west:

```
west build -b swervolf_nexys
```

**Note:** If you had to download the veervolf library above use the command:

```
west build -b veerwolf_nexys
```

### A.2.3 Executing a program on RVfpgaNexys using the Nexys A7 board with OpenOCD

1. Copy the bitstream and the binary files into the Raspberry Pi board.

2. Copy the directory ConfigFiles located in the Thesis repository: rvfpga/OtherSources/ConfigFiles
3. Download the bitstream into the fpga using OpenOCD:

```
riscv-openocd -c "set BITFILE rvfpganexys.bit" -f \
ConfigFiles/swervolf_nexys_program.cfg
```

4. Connect OpenOCD to the SoC:

```
riscv-openocd -f /rvfpga/ConfigFiles/swervolf_nexys_debug.cfg
```

5. Connect gdb and execute the application:

5.1 Open a new terminal window.

5.2 Start GDB:

```
riscv32-unknown-elf-gdb zephyr.elf
```

5.3 Execute the application:

```
target remote localhost:3333
load
continue
```